

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: 2003年 4月23日

出願番号
Application Number: 特願2003-118602
[ST. 10/C]: [JP2003-118602]

出願人
Applicant(s): 株式会社日立製作所

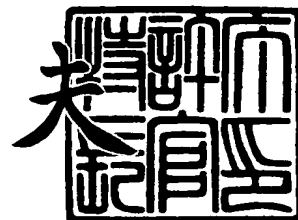
U.S. Appln. Filed 4-22-04
Inventor: m. mitsumori et al
mattingly stanger & malor
Docket NIT-421



2004年 4月 8日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫



出証番号 出証特2004-3029136

【書類名】 特許願

【整理番号】 NT03P0209

【提出日】 平成15年 4月23日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 12/08

【発明者】

 【住所又は居所】 神奈川県横浜市戸塚区戸塚町 5 0 3 0 番地 株式会社日立製作所 ソフトウェア事業部内

 【氏名】 三森 征人

【発明者】

 【住所又は居所】 神奈川県横浜市戸塚区戸塚町 5 0 3 0 番地 株式会社日立製作所 ソフトウェア事業部内

 【氏名】 中島 恵

【発明者】

 【住所又は居所】 神奈川県横浜市戸塚区戸塚町 5 0 3 0 番地 株式会社日立製作所 ソフトウェア事業部内

 【氏名】 家坂 聡

【特許出願人】

 【識別番号】 000005108

 【氏名又は名称】 株式会社日立製作所

【代理人】

 【識別番号】 100068504

 【弁理士】

 【氏名又は名称】 小川 勝男

 【電話番号】 03-3661-0071

【選任した代理人】

【識別番号】 100086656

【弁理士】

【氏名又は名称】 田中 恭助

【電話番号】 03-3661-0071

【選任した代理人】

【識別番号】 100094352

【弁理士】

【氏名又は名称】 佐々木 孝

【電話番号】 03-3661-0071

【手数料の表示】

【予納台帳番号】 081423

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 不正メモリアクセス検知方法及びそのプログラム

【特許請求の範囲】

【請求項 1】

独自のメモリ管理機能を有する言語システムと、前記言語システムの制御下で実行され前記言語システムが確保した第 1 のメモリ領域にアクセスする第 1 のプログラムコードと、OS の制御下で直接実行され前記 OS が確保した第 2 のメモリ領域にアクセスする第 2 のプログラムコードとが実行されるコンピュータにおいて、前記第 2 のプログラムコードによる前記第 1 のメモリ領域への不正なメモリアクセスを検知するための前記言語システムによって実行される方法であって、

前記言語システムは、前記第 1 のプログラムコードが前記第 2 のプログラムコードを呼び出す前に前記第 1 のメモリ領域についてメモリプロテクションを設定し、前記第 2 のプログラムコードを呼び出して実行させ、メモリプロテクション例外が発生したとき前記第 2 のプログラムコードによる不正メモリアクセスを外部に報告し、前記第 2 のプログラムコードの実行が終了して前記言語システムに制御が戻ったときに前記メモリプロテクションを解除することを特徴とする不正メモリアクセス検知方法。

【請求項 2】

前記言語システムは、さらに前記メモリプロテクション例外が発生したとき、前記第 1 のプログラムコードによる前記第 1 のメモリ領域への正常なメモリアクセスであることを検出した場合には、前記メモリプロテクションを解除し、前記正常なメモリアクセスを許し、再び前記メモリプロテクションを設定することを特徴とする請求項 1 記載の不正メモリアクセス検知方法。

【請求項 3】

前記第 1 のプログラムコードがマルチスレッド制御下で実行される場合に、前記言語システムは、あるスレッドが前記第 2 のプログラムコードを呼び出す間、他のスレッドの実行を停止させることを特徴とする請求項 1 記載の不正メモリアクセス検知方法。

【請求項 4】

独自のメモリ管理機能を有する言語システムと、前記言語システムの制御下で実行され前記言語システムが確保した第 1 のメモリ領域にアクセスする第 1 のプログラムコードと、OS の制御下で直接実行され前記 OS が確保した第 2 のメモリ領域にアクセスする第 2 のプログラムコードとが実行されるコンピュータにおいて、前記第 2 のプログラムコードによる前記第 1 のメモリ領域への不正なメモリアクセスを検知するための前記言語システムによって実行される方法であって、

前記言語システムは、前記第 1 のプログラムコードが前記第 2 のプログラムコードを呼び出す前に前記第 1 のメモリ領域についてその内容に対応するコード情報を保存し、前記第 2 のプログラムコードを呼び出して実行させ、前記第 2 のプログラムコードの実行が終了して前記言語システムに制御が戻ったときに前記第 1 のメモリ領域の内容に対応するコード情報が保存されたコード情報と一致するか否かを判定し、不一致の場合に前記第 2 のプログラムコードによる不正メモリアクセスを外部に報告することを特徴とする不正メモリアクセス検知方法。

【請求項 5】

前記言語システムは、さらに前記第 2 のプログラムコードの呼び出し中に前記第 1 のプログラムコードによる前記第 1 のメモリ領域への正常なメモリ更新を検出したとき、更新された前記第 1 のメモリ領域の内容に対応するコード情報によって前記保存されたコード情報を更新することを特徴とする請求項 4 記載の不正メモリアクセス検知方法。

【請求項 6】

前記第 1 のプログラムコードがマルチスレッド制御下で実行される場合に、前記言語システムは、あるスレッドが前記第 2 のプログラムコードを呼び出す間、他のスレッドの実行を停止させることを特徴とする請求項 4 記載の不正メモリアクセス検知方法。

【請求項 7】

独自のメモリ管理機能を有する言語システムと、前記言語システムの制御下で実行され前記言語システムが確保した第 1 のメモリ領域にアクセスする第 1 のプ

プログラムコードと、OSの制御下で直接実行され前記OSが確保した第2のメモリ領域にアクセスする第2のプログラムコードとが実行されるコンピュータにおいて、前記コンピュータに前記第2のプログラムコードによる前記第1のメモリ領域への不正なメモリアccessを検知する前記言語システムの機能を実現させるためのプログラムであって、

前記コンピュータに、前記第1のプログラムコードが前記第2のプログラムコードを呼び出す前に前記第1のメモリ領域についてメモリプロテクションを設定する機能、前記第2のプログラムコードを呼び出して実行させる機能、メモリプロテクション例外が発生したとき前記第2のプログラムコードによる不正メモリアccessを外部に報告する機能、および前記第2のプログラムコードの実行が終了して前記言語システムに制御が戻ったときに前記メモリプロテクションを解除する機能を実現させるためのプログラム。

【請求項8】

前記コンピュータに、さらに前記メモリプロテクション例外が発生したとき、前記第1のプログラムコードによる前記第1のメモリ領域への正常なメモリアccessであることを検出した場合には、前記メモリプロテクションを解除する機能、前記正常なメモリアccessを許す機能、および再び前記メモリプロテクションを設定する機能を実現させることを特徴とする請求項7記載のプログラム。

【請求項9】

前記第1のプログラムコードがマルチスレッド制御下で実行される場合に、前記コンピュータに、あるスレッドが前記第2のプログラムコードを呼び出す間、他のスレッドの実行を停止させる機能を実現させることを特徴とする請求項7記載のプログラム。

【請求項10】

独自のメモリ管理機能を有する言語システムと、前記言語システムの制御下で実行され前記言語システムが確保した第1のメモリ領域にアクセスする第1のプログラムコードと、OSの制御下で直接実行され前記OSが確保した第2のメモリ領域にアクセスする第2のプログラムコードとが実行されるコンピュータにおいて、前記コンピュータに前記第2のプログラムコードによる前記第1のメモリ

領域への不正なメモリアクセスを検知する前記言語システムの機能を実現させるためのプログラムであって、

前記コンピュータに、前記第1のプログラムコードが前記第2のプログラムコードを呼び出す前に前記第1のメモリ領域についてその内容に対応するコード情報を保存する機能、前記第2のプログラムコードを呼び出して実行させる機能、前記第2のプログラムコードの実行が終了して前記言語システムに制御が戻ったときに前記第1のメモリ領域の内容に対応するコード情報が保存されたコード情報と一致するか否かを判定する機能、および不一致の場合に前記第2のプログラムコードによる不正メモリアクセスを外部に報告する機能を実現させるためのプログラム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、不正メモリアクセス検知方法に係り、特に不正なメモリアクセスが発生しないシステムと、自由にメモリアクセスを行えるシステムとが混在する環境において好適な不正メモリアクセス検知方法に関する。

【0002】

【従来の技術】

オブジェクト指向言語として一般的に知られている J a v a プログラムを実行する環境である J a v a V M は、J a v a プログラムを実行する際に使用するメモリ領域の管理を独自に行っており、J a v a プログラムが実行されている限りにおいては不正なメモリアクセスが発生しないシステムとなっている（J a v a は、米国 S u n M i c r o s y s t e m s , I n c . の登録商標である）。しかし J a v a プログラムを実行する上で必要に応じて呼び出される他言語（例えば C 言語）で作成されたプログラムの実行中は、O S がメモリ領域の管理を行っているため、その間は J a v a V M では不正なメモリアクセスが発生したかどうかの検出ができない。そのため J a v a プログラムから呼び出された他言語で作成されたプログラムが誤って J a v a V M の管理しているメモリ領域を不正にアクセスして更新してしまう可能性がある。しかしこのような不正メモリアクセ

スを早期に検出する技術は知られていない。

【0003】

なおこの種の技術として関連するものには、例えば特開平6-44129号公報（特許文献1）、特開平5-28053号公報（特許文献2）などがある。

【0004】

【特許文献1】

特開平6-44129号公報

【特許文献2】

特開平5-28053号公報

【0005】

【発明が解決しようとする課題】

上記の従来技術は、J a v a プログラムを実行する上で必要に応じて呼び出される他言語で作成されたプログラムの実行時に、他言語プログラムがJ a v a V Mで管理しているメモリ領域を不正にアクセスして更新した場合、その後に実行されるJ a v a プログラムがそのメモリ領域にアクセスし異常状態となるまでは不正なメモリアクセスが発生していたことを検知できず、問題のあるプログラムを特定することが困難であった。

【0006】

本発明の目的は、不正なメモリアクセスが発生しないシステムで動作するプログラムから呼ばれた自由にメモリアクセスを行えるシステムで動作するプログラムが不正なメモリアクセスをしたことを早期に検出することにある。

【0007】

【課題を解決するための手段】

本発明は、不正なメモリアクセスが発生しないシステムで動作しているプログラムから呼ばれた自由にメモリアクセスが行えるシステムで動作するプログラムが実行中又は実行後早期に不正なメモリアクセスが発生したことを検出する技術を特徴とする。

【0008】

【発明の実施の形態】

不正なメモリアクセスが発生しないシステムとして J a v a VM を例とし、そのシステムで動作するプログラムとして J a v a で記述したプログラムを例とし、自由にメモリアクセスを行えるシステムで動作するプログラムとして C 言語で記述したネイティブメソッドライブラリを使用する場合を例とする実施の形態について以下図面を用いて説明する。

【0009】

図1は、実施形態の J a v a VM の構成とその入力ファイルを示す図である。101は記憶装置上に格納されている J a v a ソースプログラムである。102は J a v a ソースプログラムを J a v a VM で実行できるように中間言語で記述されるバイトコードに変換する J a v a コンパイラである。103は J a v a コンパイラで作成したバイトコードが格納されている J a v a クラスファイルである。104はバイトコードを実行する際に必要な他のバイトコード、すなわちクラスファイルを収めたクラスライブラリである。105はバイトコードから呼び出される J a v a 言語以外の言語で記述されたネイティブメソッドライブラリである。106は本発明を実施する言語システムである J a v a VM の本体である。107は J a v a クラスファイル103のバイトコードをメモリにロードするバイトコード読み取り部である。108は入力されたバイトコードやネイティブメソッドライブラリ105を呼び出して、J a v a プログラムを実際に実行する実行部である。実行部108は、バイトコードを実行する際に必要な他のバイトコードを収めたクラスファイルをロードするクラスライブラリロード部109、J a v a 言語以外の言語で記述されたネイティブメソッドライブラリ105をメモリにロードするネイティブメソッドライブラリロード部110、J a v a プログラムを実行する際に J a v a VM が必要なメモリ領域を確保するメモリ確保部111、バイトコードを実行するバイトコード実行部112、ネイティブメソッドライブラリ105を実行するネイティブメソッドライブラリ実行部113、ネイティブメソッドライブラリの実行中、または実行後に不正なメモリアクセスが発生したかどうかを検出する不正メモリアクセス検出部114から構成される。

【0010】

なお図示していないが、J a v a VM 106はOS（オペレーティングシステ

ム)の制御を受ける。このOSはネイティブのメモリ管理機能を有する。ネイティブメソッドライブラリは、このOSのもつメモリ管理機能を利用して自身のメモリ領域を確保する。言うまでもないが、図1のJavaコンパイラ102、JavaVM106及びOSは、CPU、メモリ、記憶装置、入力装置、表示装置などをもつコンピュータのCPUによって実行されるプログラムである。Javaソースプログラム101、Javaクラスファイル103、クラスライブラリ104及びネイティブメソッドライブラリ105は、この記憶装置に格納されるプログラムコードである。Javaクラスファイル103、クラスライブラリ104及びネイティブメソッドライブラリ105は、このコンピュータによって実行されるプログラムコードである。

【0011】

入力装置などからコマンドが投入されることによってJavaVM106が起動され、JavaVM106が実行開始される。バイトコード読み取り部107は、Javaクラスファイル103からバイトコードを読み込むと、実行部108に制御を移し、バイトコード実行部112が読み込まれたバイトコードを実行する。

【0012】

図2は、実行部108のうち本発明に係る部分の処理手順を示すフローチャートである。ステップ201及び202は、バイトコードを実行する際に必要なクラスライブラリ104をメモリにロードするクラスライブラリロード部109の処理である。ステップ203及び204は、バイトコードを実行する際に必要なネイティブメソッドライブラリ105をメモリにロードするネイティブメソッドライブラリロード部110の処理である。ステップ205は、入力されたバイトコードをJavaVM内で実行する際に必要なメモリを確保するメモリ確保部111の処理である。ステップ206は、実際にバイトコードを実行するバイトコード実行部112の処理である。ステップ207及び208は、ネイティブメソッドライブラリを実行するネイティブメソッドライブラリ実行部113の処理である。ステップ209及び210は、ネイティブメソッドライブラリを実行したとき、不正なメモリアクセスが発生したかどうかを検出する不正メモリアク

セス検出部 114 の処理である。

【0013】

バイトコードが入力されたとき、クラスライブラリロード部 109 は、ステップ 201 で他にも必要なバイトコードがあるかどうかを判断する。他にも必要なバイトコードがある場合は、ステップ 202 でバイトコードを収めたクラスライブラリ 104 からクラスライブラリをメモリにロードする。

【0014】

ネイティブメソッドライブラリロード部 110 は、ステップ 203 で入力されたバイトコードに、Java 言語以外の言語で記述されたネイティブメソッドライブラリを呼び出す処理があるかどうかを判断する。ネイティブメソッドライブラリを呼び出す処理がある場合は、ステップ 204 でネイティブメソッドライブラリ 105 からネイティブメソッドライブラリをメモリにロードする。

【0015】

メモリ確保部 111 は、Java VM 106 が独自に有するメモリ管理機能を利用して、ステップ 205 で Java VM 内でバイトコードを実行する際に必要なメモリを確保する。Java VM 106 は、将来不正なメモリアクセスが発生した場合でも検知できるように、確保したメモリ領域を全てメモリ管理テーブルに登録する。またメモリ確保部 111 は、不必要になったメモリを集める処理も行う。

【0016】

バイトコード実行部 112 は、ステップ 206 で実際にバイトコードを実行する。ネイティブメソッドライブラリ実行部 113 は、ステップ 207 で現在実行しているバイトコードから Java 言語以外の言語で記述されたネイティブメソッドライブラリを呼び出すか判断する。ネイティブメソッドライブラリを呼び出す場合は、ステップ 208 でネイティブメソッドライブラリを呼び出し、呼び出したネイティブメソッドライブラリに実行の制御を渡す。

【0017】

不正メモリアクセス検出部 114 は、ステップ 209 でネイティブメソッドライブラリの実行中、または実行後に不正なメモリアクセスが発生したかどうかを

検出する。不正なメモリアクセスが発生した場合は、ステップ 2 1 0 で該当するネイティブメソッドライブラリを外部に報告する。不正メモリアクセス検出部 1 1 4 は、エラーメッセージを表示装置に表示するか又は指定されたファイルに出力する。以上のように実行部 1 0 8 の処理が終了すると、制御は再びバイトコード読み取り部 1 0 7 に戻される。

(1) 実施例 1

図 3 は、OS がメモリ保護機能をもちかつマルチスレッド制御を行うシステムにおいて、ステップ 2 0 8 から 2 1 0 の処理を実装する場合の処理のフローチャートである。ネイティブメソッドライブラリ実行部 1 1 3 は、ステップ 3 0 1 でこれから呼び出すネイティブメソッドライブラリの処理の中で、不正なメモリアクセスが発生しても検知できるように、ステップ 2 0 5 の処理で確保した J a v a V M 内で使用しているメモリ領域に書き込み禁止のプロテクトをかける。ネイティブメソッドライブラリ実行部 1 1 3 は、ステップ 3 0 2 でネイティブメソッドライブラリを呼び出す。

【 0 0 1 8 】

ネイティブメソッドライブラリ実行中に、プロテクトがかけられたメモリ領域がアクセスされてメモリプロテクション例外が発生したとき、制御は J a v a V M 1 0 6 に戻される。プロテクトがかけられたメモリ領域をアクセスしたスレッドが J a v a V M 内で動作しているスレッドの場合には、J a v a V M 1 0 6 の例外処理プログラム（不正メモリアクセス検出部 1 1 4）は、ステップ 3 0 3 で一旦メモリ領域のプロテクトを外し、ステップ 3 0 4 でプロテクトを外したメモリ領域について正常な更新を行う。ステップ 3 0 5 で再度メモリ領域にプロテクトをかける処理を行う。ステップ 3 0 3 ～ 3 0 5 の処理は、他のスレッドがメモリ領域をアクセスしないようにアトミックに行われる。その後、そのまま元の処理に戻って処理を続行する。またプロテクトがかけられたメモリ領域をアクセスしたスレッドがネイティブメソッドライブラリのスレッドの場合は、ステップ 3 0 6 でネイティブメソッドライブラリのプログラムをエラーメッセージとして報告して処理を終了する。

【 0 0 1 9 】

例外が発生せずにネイティブメソッドライブラリの実行が終了し、J a v a V M 1 0 6 に制御が戻ってきたとき、ネイティブメソッドライブラリ実行部 1 1 3 は、ステップ 3 0 7 でメモリにかけたプロテクトを外す。

【0020】

もし J a v a V M 内で使用しているメモリ領域にプロテクトをかけない指示があった場合は、ステップ 3 0 1、3 0 3、3 0 5、3 0 7 の処理を実行しない。すなわちネイティブメソッドライブラリが不正なメモリアクセスをしないことが確認されたとき、オーバヘッドとなる処理を除去することができる。

【0021】

図 4 は、O S がメモリ保護機能をもちかつマルチスレッド制御を行うシステムで、ステップ 3 0 2 で呼び出したネイティブメソッドライブラリの実行中に、不正なメモリアクセスが発生したときの例を示す図である。

【0022】

図 4 は、不正なメモリアクセスが発生しない J a v a V M 1 0 6 のシステムで動作する J a v a プログラムから、自由にメモリアクセスを行えるシステムで動作する C 言語で記述されたネイティブメソッドライブラリ 4 0 2 の処理が呼ばれた状態を示している。ネイティブメソッドライブラリ (f u n c A) 4 0 2 は、ポインタ i p のポイント先である領域 4 0 3 を更新するつもりが、誤ってライトプロテクトがかけられたメモリ領域 4 0 4 内の領域 4 0 5 を更新しようとした。この場合、更新しようとしたスレッドがネイティブメソッドライブラリで動作しているスレッドのためメモリプロテクション例外が発生し、J a v a V M 1 0 6 の例外処理プログラムは、その時に実行していたネイティブメソッドライブラリ (f u n c A) を報告して終了する。もしライトプロテクトがかけられたメモリ領域 4 0 4 内の領域 4 0 6 を更新しようとしたスレッドが J a v a V M で動作しているスレッドの場合は、J a v a V M 1 0 6 は、そのプロテクトを外して領域 4 0 6 の更新を許し、再びメモリ領域 4 0 4 にライトプロテクトをかける。

(2) 実施例 2

図 5 は、O S がメモリ保護機能をもたずかつマルチスレッド制御を行うシステムにおいて、ステップ 2 0 8 から 2 1 0 の処理を実装する場合の処理のフローチ

ャートである。ネイティブメソッドライブラリ実行部 113 は、ステップ 501 でステップ 205 の処理で確保した J a v a VM 内で使用しているメモリ領域についてその内容のチェックサムを求めて何かの記憶領域に退避する。この処理は、他のスレッドがメモリ領域を更新してチェックサムも更新しないようにアトミックに行われる。

【0023】

複数のスレッドがネイティブメソッドライブラリを実行している時に、不正なメモリアクセスが発生した場合、どのスレッドで実行しているネイティブメソッドライブラリに問題があるのか特定できなくなる。この状態を避けるためにネイティブメソッドライブラリ実行部 113 は、ステップ 502 で他の J a v a VM のスレッドがネイティブメソッドライブラリ 105 を実行している場合は、そのスレッドが実行しているネイティブメソッドライブラリ 105 の実行が終わるまで待つ。その後ネイティブメソッドライブラリ実行部 113 は、ステップ 503 でネイティブメソッドライブラリを呼び出す。

【0024】

J a v a VM 内で動作しているスレッドがメモリを更新する場合は、ステップ 504 で本来のメモリ更新を許し、ステップ 505 で J a v a VM のスレッドが更新した部分のみの更新前後の差分を計算し、新しいチェックサムによってステップ 501 で退避したチェックサムを更新する。この処理は、他のスレッドがメモリ領域を更新してチェックサムも更新しないようにアトミックに行われる。

【0025】

J a v a VM 内で動作しているスレッドがメモリを更新する時に、他の J a v a VM のスレッドがネイティブメソッドライブラリを呼び出していない場合は、ステップ 505 の処理を行う必要はない。ネイティブメソッドライブラリのスレッドがメモリ領域を更新する場合は、不正なメモリアクセスがあってもそのまま処理を続行する。

【0026】

ネイティブメソッドライブラリから処理が戻ってきたとき、ネイティブメソッドライブラリ実行部 113 は、ステップ 506 でステップ 205 の処理で確保し

た J a v a VM内で使用しているメモリ領域の内容について現在のチェックサムを求める処理をアトミックに行う。不正メモリアクセス検出部 114 は、ステップ 507 で退避しておいたチェックサムとステップ 506 で求めたチェックサムとを比較する。両者が不一致の場合は、ステップ 508 で直前に呼び出したネイティブメソッドライブラリをエラーメッセージとして外部に報告して処理を終了する。

【0027】

もし J a v a VM内で使用しているメモリ領域のチェックサムを求めない指示があった場合は、ステップ 501、502、505 から 508 の処理は実行しない。

【0028】

図 6 は、OS がメモリ保護機能をもたずかつマルチスレッド制御を行うシステムで、ステップ 503 で呼び出したネイティブメソッドライブラリの実行中に、不正なメモリアクセスが発生したときの例を示す図である。

【0029】

図 6 は、不正なメモリアクセスが発生しない J a v a VM106 のシステムで動作する J a v a プログラムから、自由にメモリアクセスを行えるシステムで動作する C 言語で記述されたネイティブメソッドライブラリ 402 の処理が呼ばれた状態を示している。ネイティブメソッドライブラリ実行部 113 は、ネイティブメソッドライブラリ 402 を呼ぶ前に J a v a VM内で使用しているメモリ領域 404 内の領域 606 にチェックサムを退避する（ステップ 501）。ネイティブメソッドライブラリ（f u n c A）402 は、ポインタ i p のポイント先である領域 403 を更新するつもりが、誤って J a v a VM内で使用しているメモリ領域 404 内の領域 405 を更新して、たまたま処理が正常に終了した。この場合、制御は再びネイティブメソッドライブラリ 402 の処理を呼び出した J a v a プログラムの呼び出し元に戻ってくる。その直後、J a v a VM内で使用しているメモリ領域 404 のチェックサムを求める処理（ステップ 506）を実行し、求めたチェックサムと領域 606 に退避しておいたチェックサムの比較を行う（ステップ 507）。J a v a VM内で使用しているメモリ領域 404 内の領

域 405 が不正に更新されているため比較結果が不一致となり、不正メモリアクセス検出部 114 は、直前に呼ばれたネイティブメソッドライブラリ 402 に問題があることを報告して終了する。もし Java VM で動作しているスレッドがチェックサムによってライトプロテクトがかけられたメモリ領域 404 の領域 406 を更新した場合は、領域 406 の更新前後の差分を求め、領域 606 に退避してあるチェックサムの値を更新する（ステップ 505）。

【0030】

なおチェックサムの値を格納する領域 606 は、メモリ領域 404 内に限られず、任意のメモリ又は記憶装置でよい。

（3）実施例 3

図 7 は、OS がメモリ保護機能をもちかつマルチスレッド制御を行うシステムにおいて、Java VM からネイティブメソッドライブラリを呼び出す時に、Java VM 内で動作している他のスレッドを停止させることができる場合のステップ 208～210 を実装する処理のフローチャートである。実行部 108 は、ステップ 701 でこれからライトプロテクトをかける Java VM 内で使用しているメモリ領域を Java VM で起動している他のスレッドがアクセスしないように、現在起動している Java VM 内の他のスレッドの実行を停止させる。実行部 108 は、ステップ 702 でこれから呼び出すネイティブメソッドライブラリの処理の中で、不正なメモリアクセスが発生しても検知できるように、ステップ 205 の処理で確保した Java VM 内で使用しているメモリ領域にプロテクトをかける。

【0031】

ネイティブメソッドライブラリ実行部 113 は、ステップ 703 でネイティブメソッドライブラリを呼び出す。ネイティブメソッドライブラリ実行中に、ライトプロテクトがかけられたメモリ領域がアクセスされてメモリプロテクション例外が発生した場合、ステップ 704 で、止めた Java VM 内の他のスレッドを再開させる。メモリ領域をアクセスしたスレッドは Java VM のスレッドではないので、不正メモリアクセス検出部 114 は、ステップ 705 でその時に実行していたネイティブメソッドライブラリのプログラムを報告して処理を終了させ

る。

【0032】

正常にネイティブメソッドライブラリから処理が戻ってきたとき、実行部 1 0 8 は、ステップ 7 0 6 でメモリにかけたプロテクトを外し、ステップ 7 0 7 で、止めていた J a v a VM内の他のスレッドを再開させる。

(4) 実施例 4

図 8 は、O S がメモリ保護機能をもたずかつマルチスレッド制御を行うシステムにおいて、J a v a VMからネイティブメソッドライブラリを呼び出す時に、J a v a VM内で動作している他のスレッドを停止させることができる場合のステップ 2 0 8 ~ 2 1 0 を実装する処理のフローチャートである。実行部 1 0 8 は、ステップ 8 0 1 でこれからチェックサムを求める J a v a VM内で使用しているメモリ領域を J a v a VM内で起動している他のスレッドがアクセスしないように、現在起動している J a v a VM内の他のスレッドを全て止める。ネイティブメソッドライブラリ実行部 1 1 3 は、ステップ 8 0 2 でステップ 2 0 5 の処理で確保した J a v a VM内で使用しているメモリ領域のチェックサムを求めて何らかの記憶領域に退避する。

【0033】

ネイティブメソッドライブラリ実行部 1 1 3 は、ステップ 8 0 3 でネイティブメソッドライブラリを呼び出す。ネイティブメソッドライブラリのスレッドがメモリ領域を更新する場合は、不正なメモリ領域であってもそのまま処理は続行する。

【0034】

ネイティブメソッドライブラリから処理が戻ってきたとき、ネイティブメソッドライブラリ実行部 1 1 3 は、ステップ 8 0 4 で確保した J a v a VM内で使用しているメモリ領域の現在のチェックサムを求める。次に実行部 1 0 8 は、ステップ 8 0 5 で、止めていた J a v a VM内の他のスレッドを再開させる。

【0035】

不正メモリアクセス検出部 1 1 4 は、ステップ 8 0 6 で退避しておいたチェックサムとステップ 8 0 4 で求めたチェックサムとを比較する。両者が不一致の場

合は、不正メモリアクセス検出部 114 は、ステップ 807 で直前に呼び出したネイティブメソッドライブラリをエラーメッセージとして外部に報告して処理を終了する。

【0036】

なお上記実施例 2 及び実施例 4 では、チェックサムを計算したが、チェックサムを計算する代わりにハッシュ関数を用いたりデータ圧縮した結果を用いるなど、上記メモリ領域の内容を入力とする関数手続きによって得られる結果のコード情報であって上記メモリ領域の内容に一意又は高い確率で対応するコード情報が得られるような関数手続きであれば何でもよい。もちろん上記メモリ領域の内容をそのままメモリなどの記憶装置に退避する場合も含まれる。

【0037】

以上の実施例において、必要とする Java VM 内のスレッド管理、アトミック処理等は、従来から Java VM に備わっている機能であるため詳述しない。

【0038】

【発明の効果】

本発明によれば、不正なメモリアクセスが発生しないシステムで動作しているプログラムから呼ばれた自由にメモリアクセスを行えるシステムで動作するプログラムが実行中又は実行後早期に不正なメモリアクセスが発生したことを検出できる。

【図面の簡単な説明】

【図 1】

実施形態の Java VM の構成図である。

【図 2】

実施形態の実行部 108 の処理手順を示すフローチャートである。

【図 3】

実施例 1 の処理手順を示すフローチャートである。

【図 4】

実施例 1 における不正メモリアクセスを説明する図である。

【図 5】

実施例 2 の処理手順を示すフローチャートである。

【図 6】

実施例 2 における不正メモリアクセスを説明する図である。

【図 7】

実施例 3 の処理手順を示すフローチャートである。

【図 8】

実施例 4 の処理手順を示すフローチャートである。

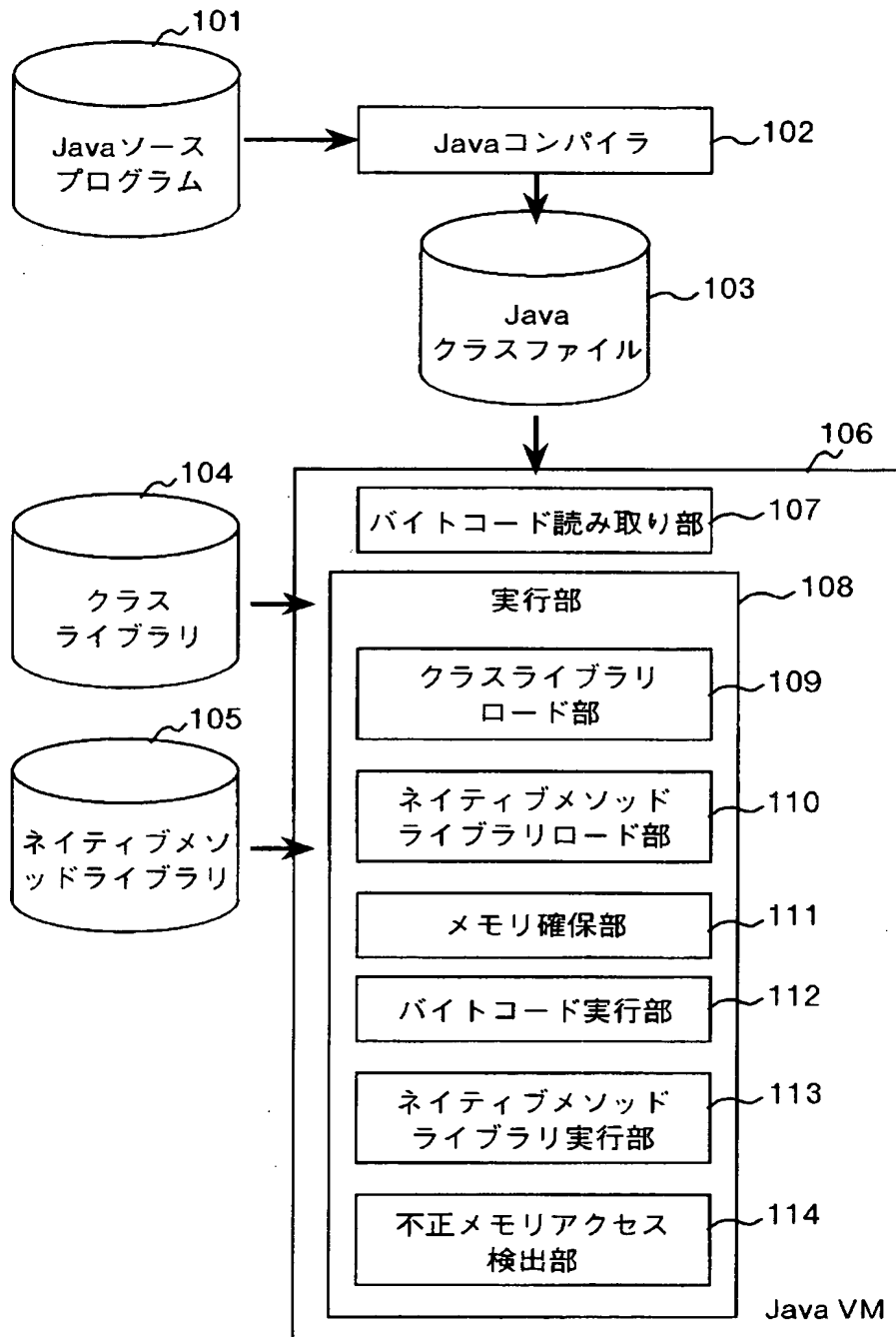
【符号の説明】

1 0 3…J a v a クラスファイル、1 0 5…ネイティブメソッドライブラリ、
1 0 6…J a v a V M、1 1 3…ネイティブメソッドライブラリ実行部、1 1 4
…不正メモリアクセス検知部。

【書類名】 図面

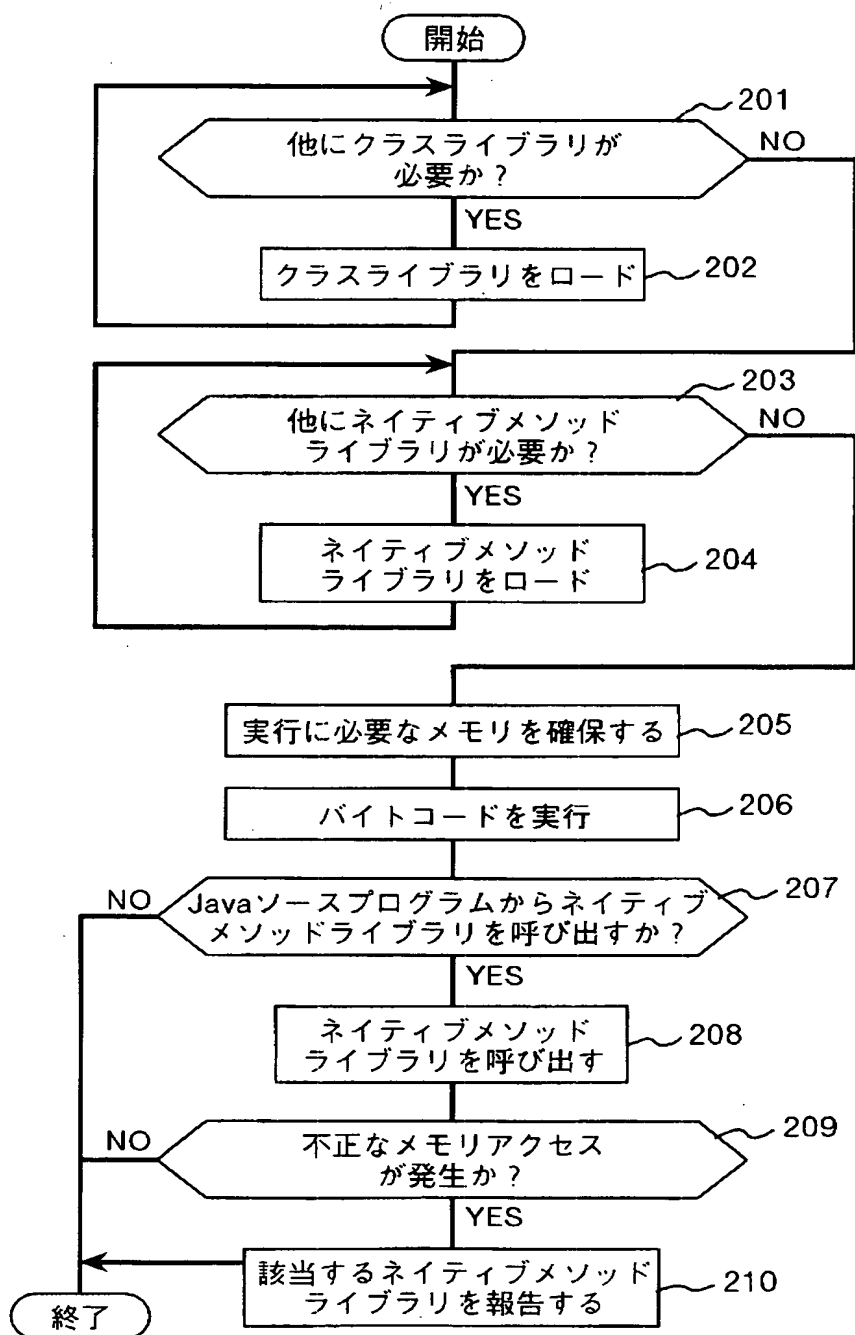
【図 1】

図 1



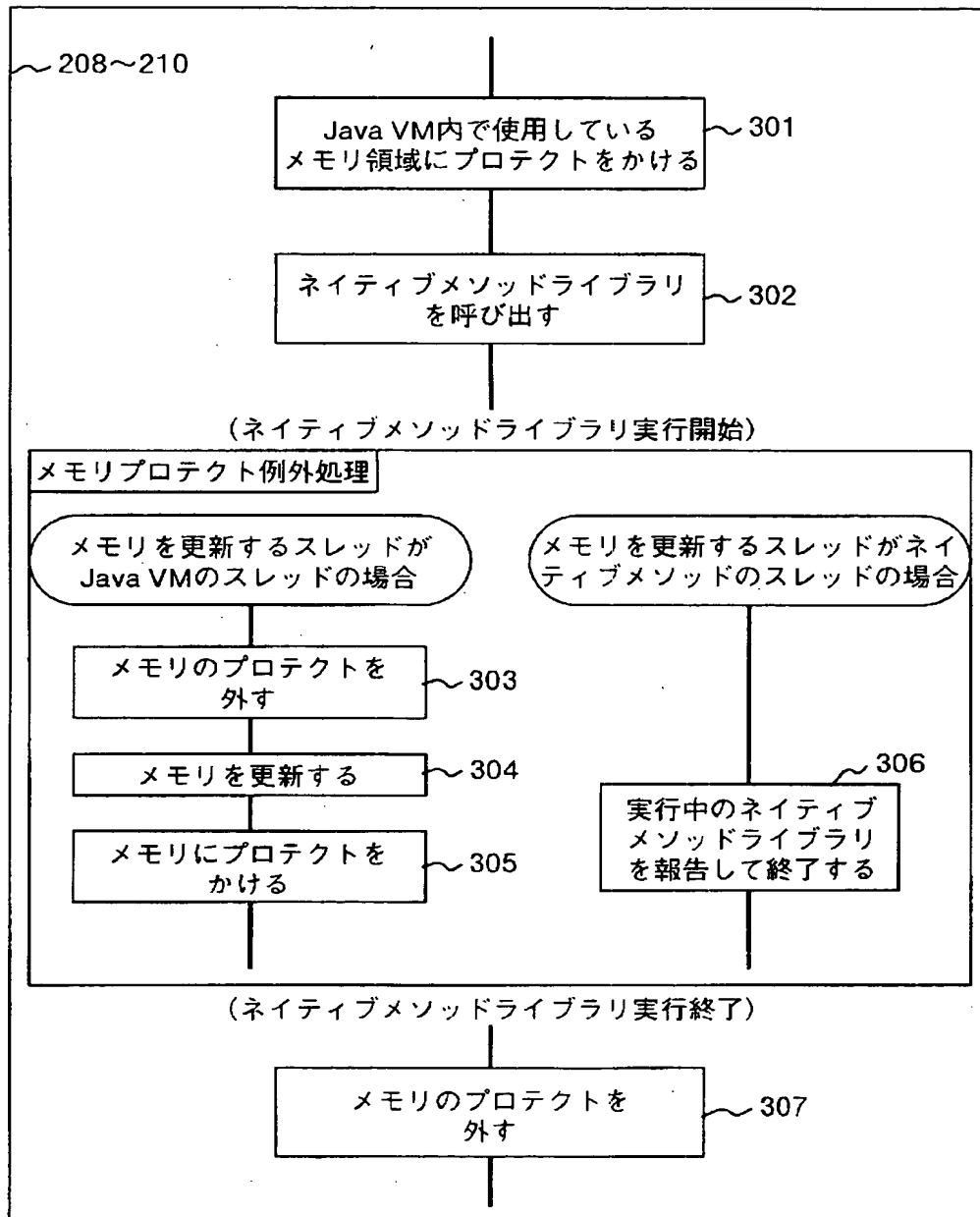
【図 2】

図 2



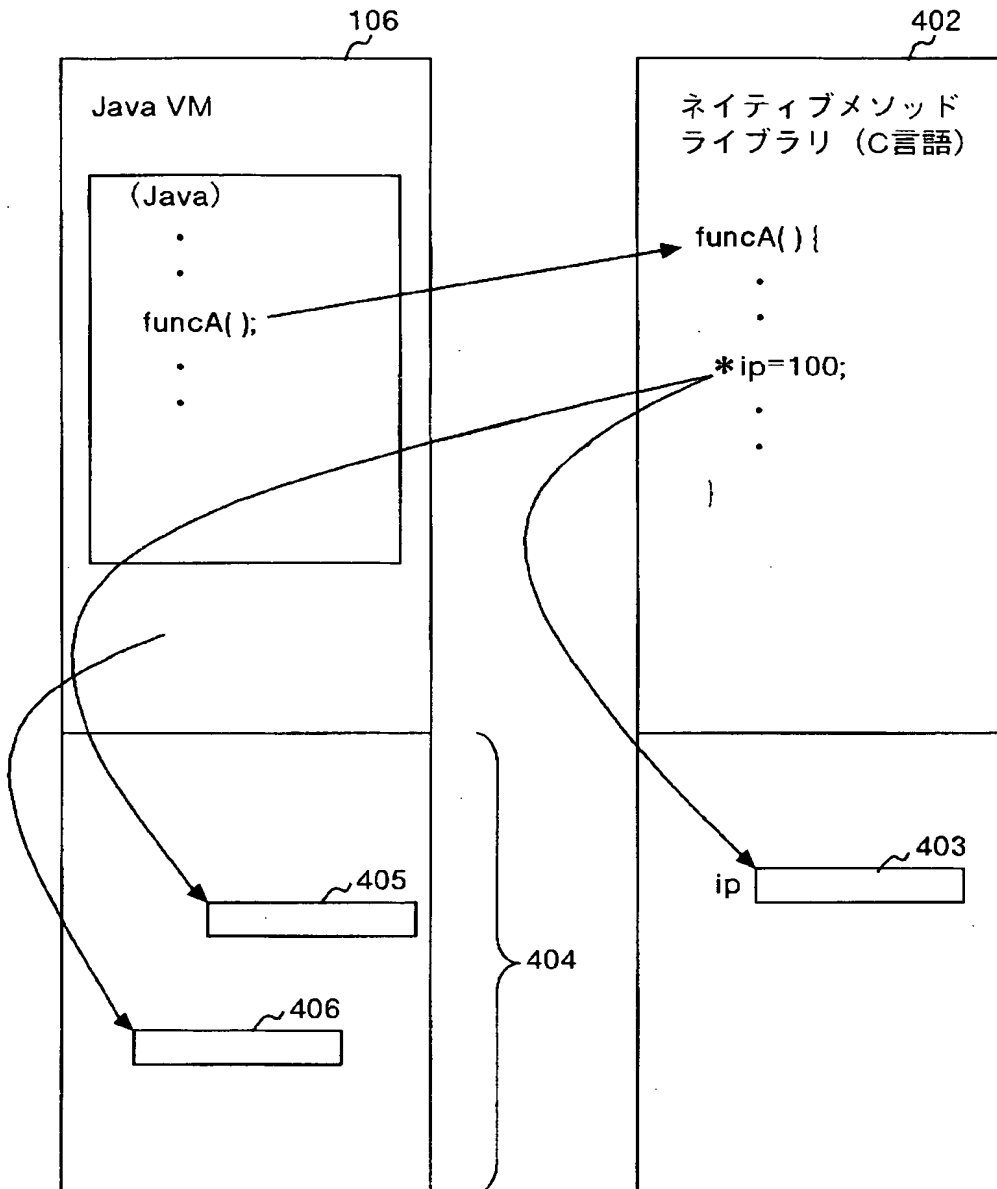
【図 3】

図 3



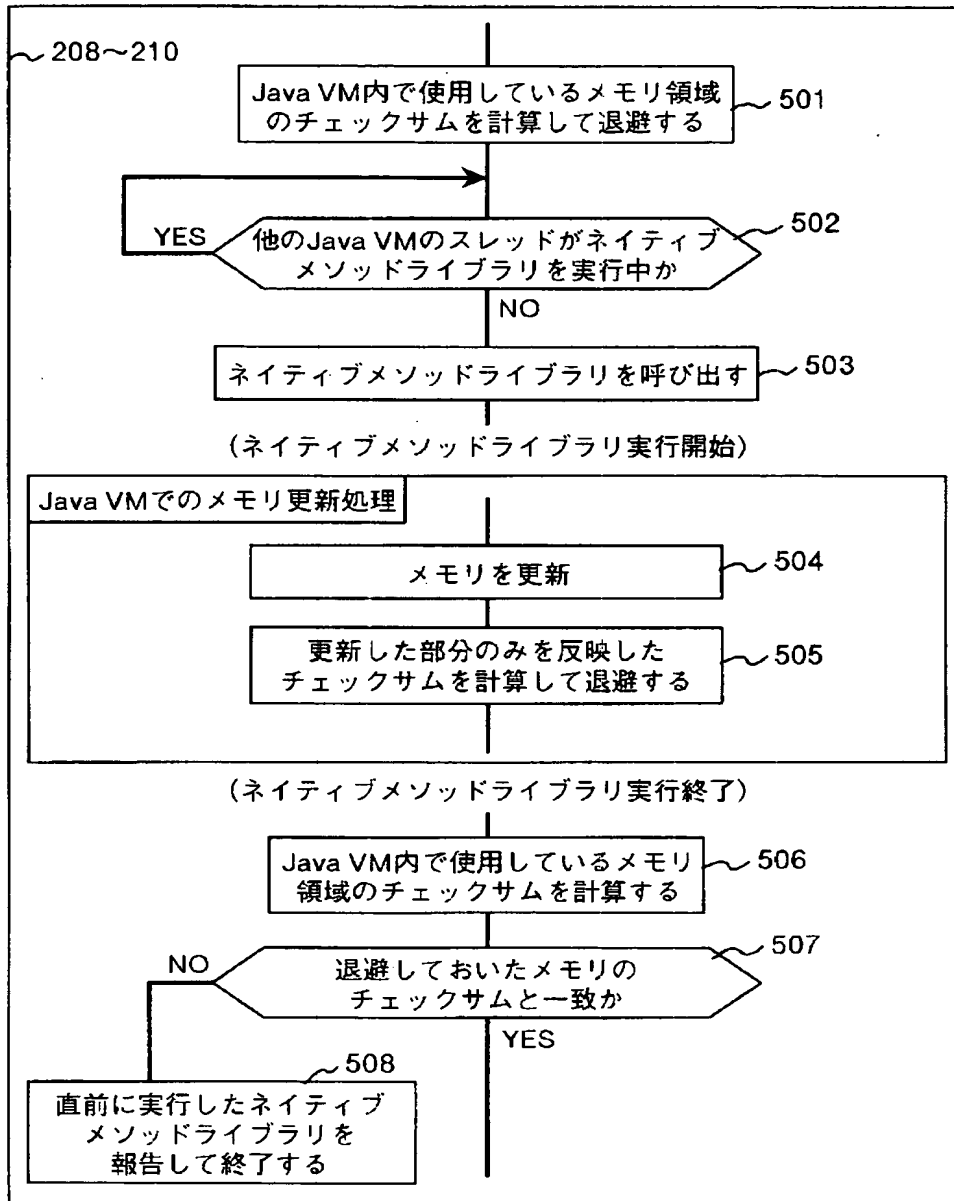
【図 4】

図 4



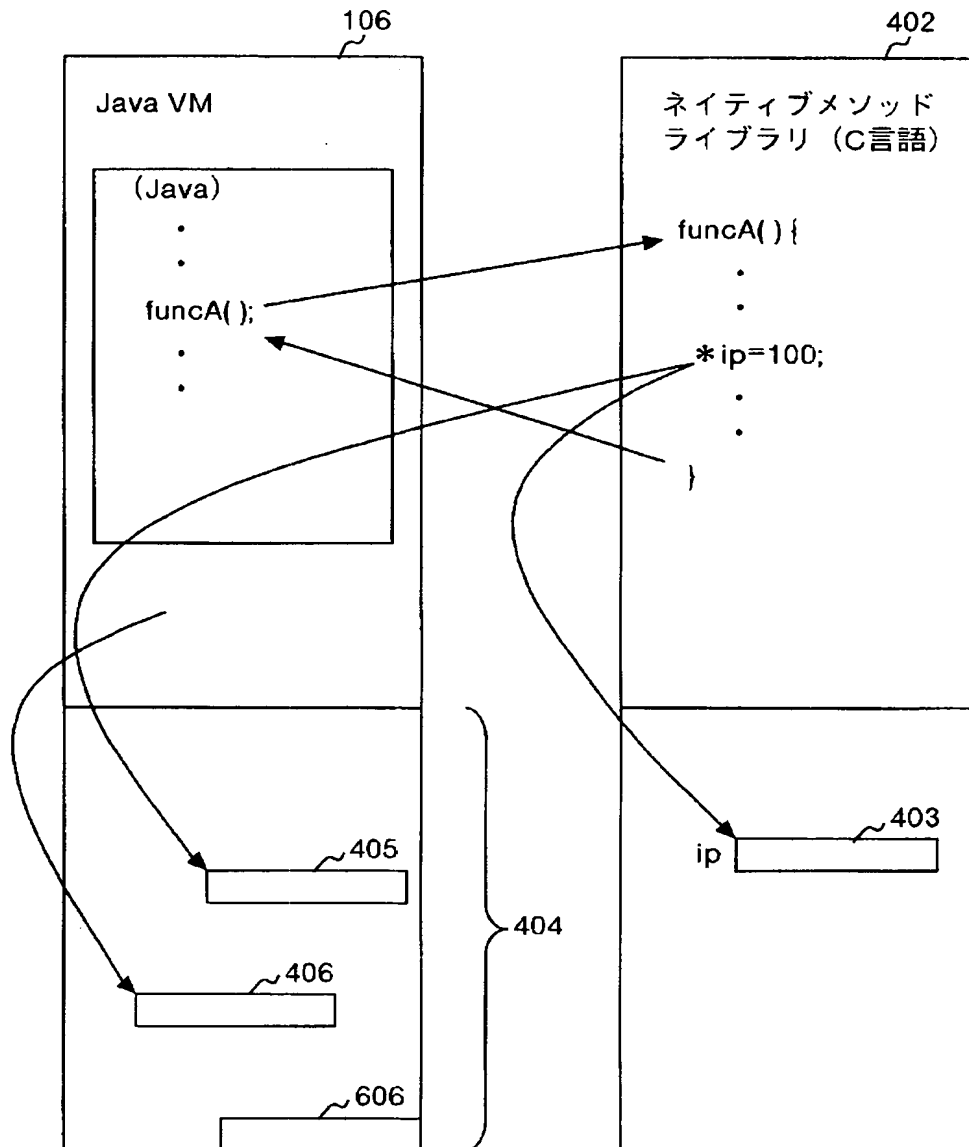
【図 5】

図 5



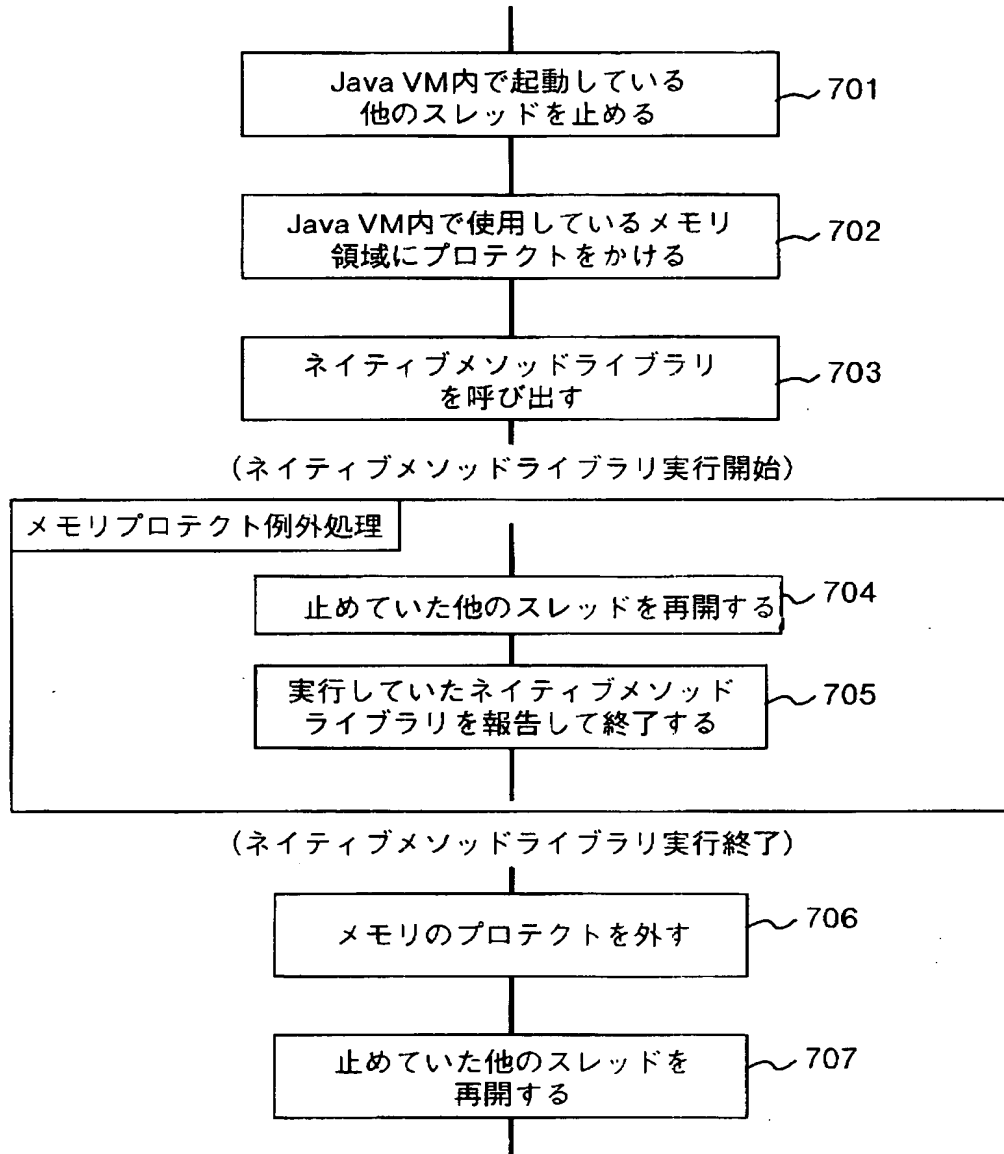
【図 6】

図 6



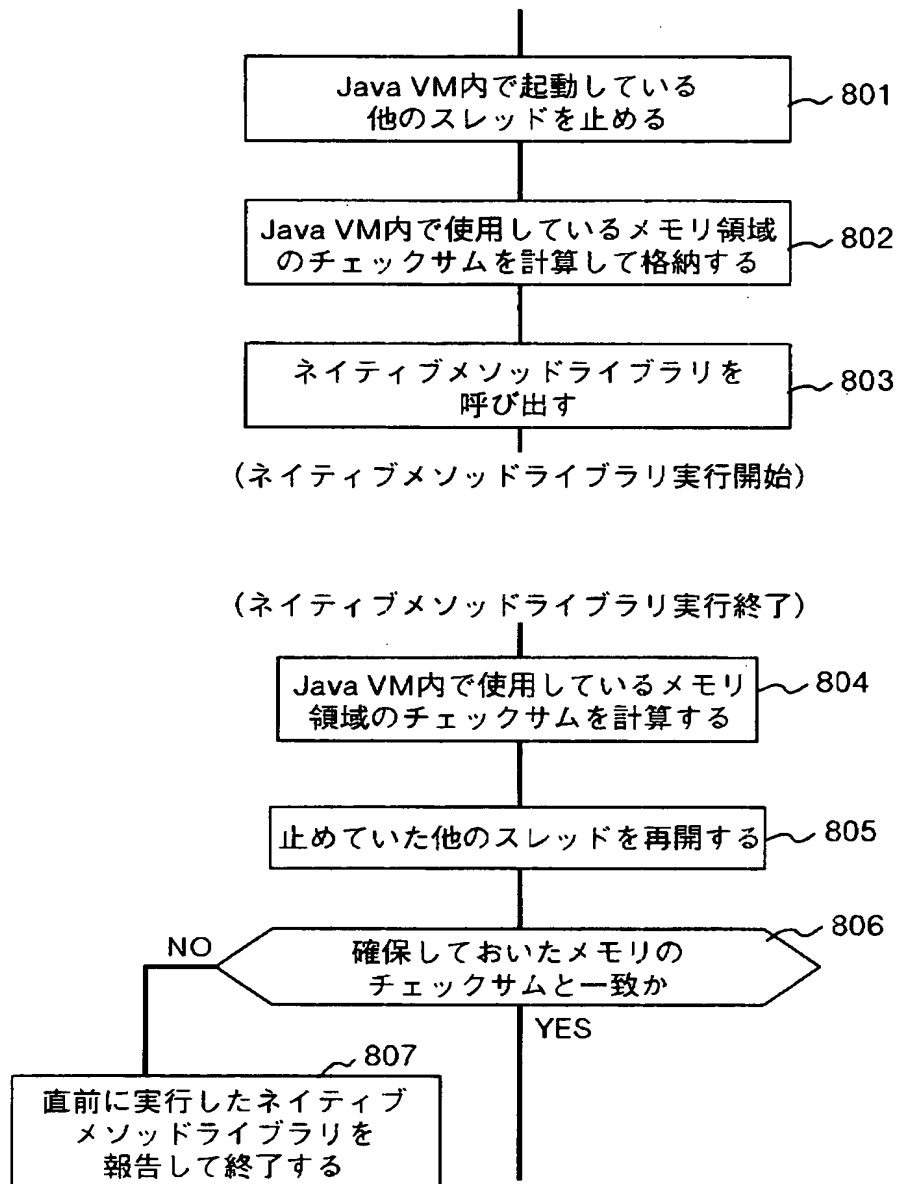
【図 7】

図 7



【図 8】

図 8



【書類名】 要約書

【要約】

【課題】 不正なメモリアクセスが発生しないシステムで動作するプログラムから呼ばれた自由にメモリアクセスを行えるシステムで動作するプログラムが不正なメモリアクセスをしたことを早期に検出する。

【解決手段】 J a v a V M 1 0 6 の実行部 1 0 8 は、読み込まれた J a v a のバイトコードを実行する。ネイティブメソッドライブラリ実行部 1 1 3 は、ネイティブメソッドライブラリを呼び出して実行させる。不正メモリアクセス検出部 1 1 4 は、ネイティブメソッドライブラリの実行中又は実行後に、メモリ確保部 1 1 1 が確保したメモリ領域についてネイティブメソッドライブラリによる不正なメモリアクセスを検出する。

【選択図】 図 1

特願 2 0 0 3 - 1 1 8 6 0 2

出 願 人 履 歴 情 報

識別番号

[0 0 0 0 0 5 1 0 8]

1. 変更年月日

1 9 9 0 年 8 月 3 1 日

[変更理由]

新規登録

住 所

東京都千代田区神田駿河台 4 丁目 6 番地

氏 名

株式会社日立製作所